

Fig. 1A

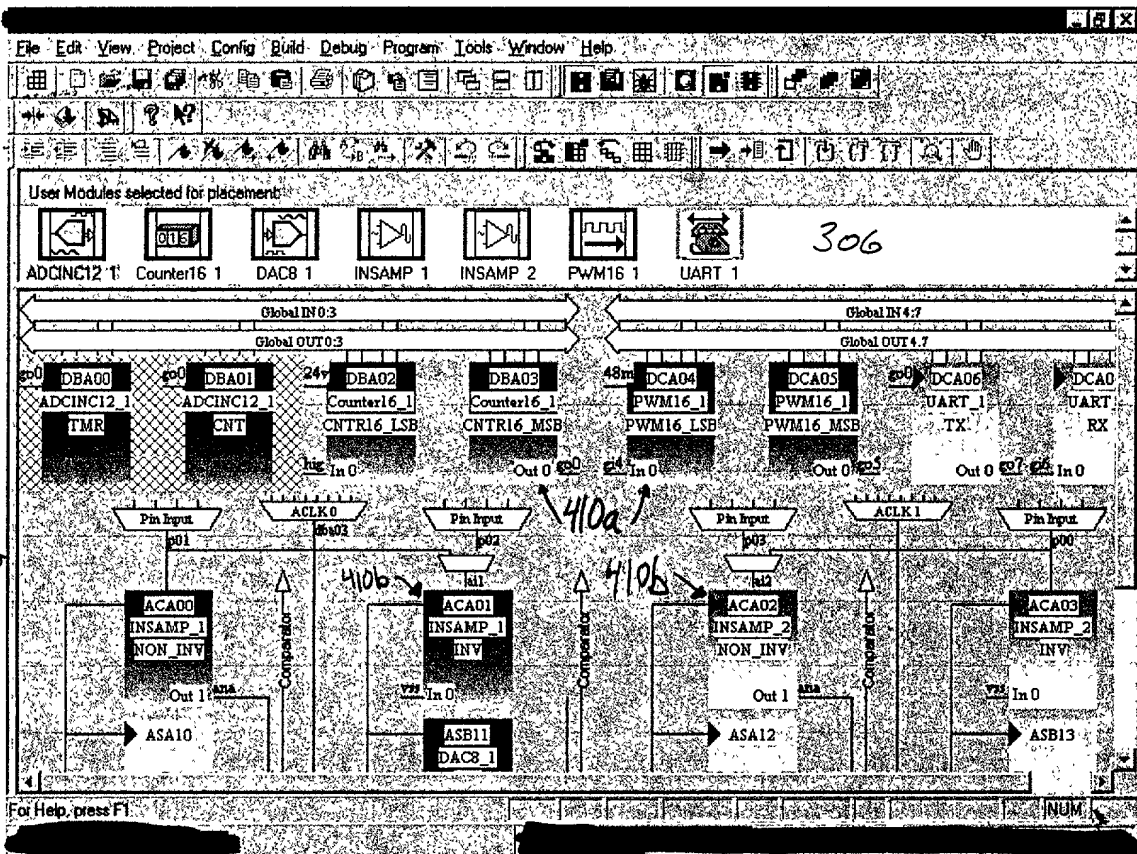


Fig 1 B

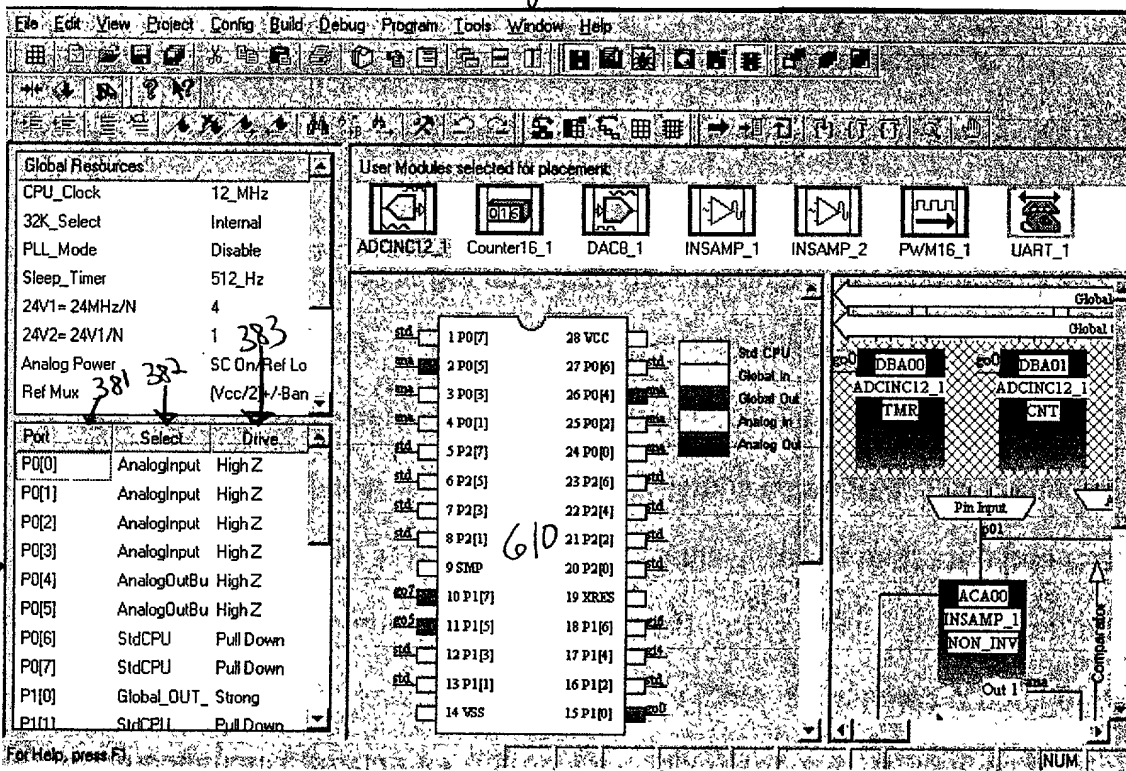


Fig 1 C

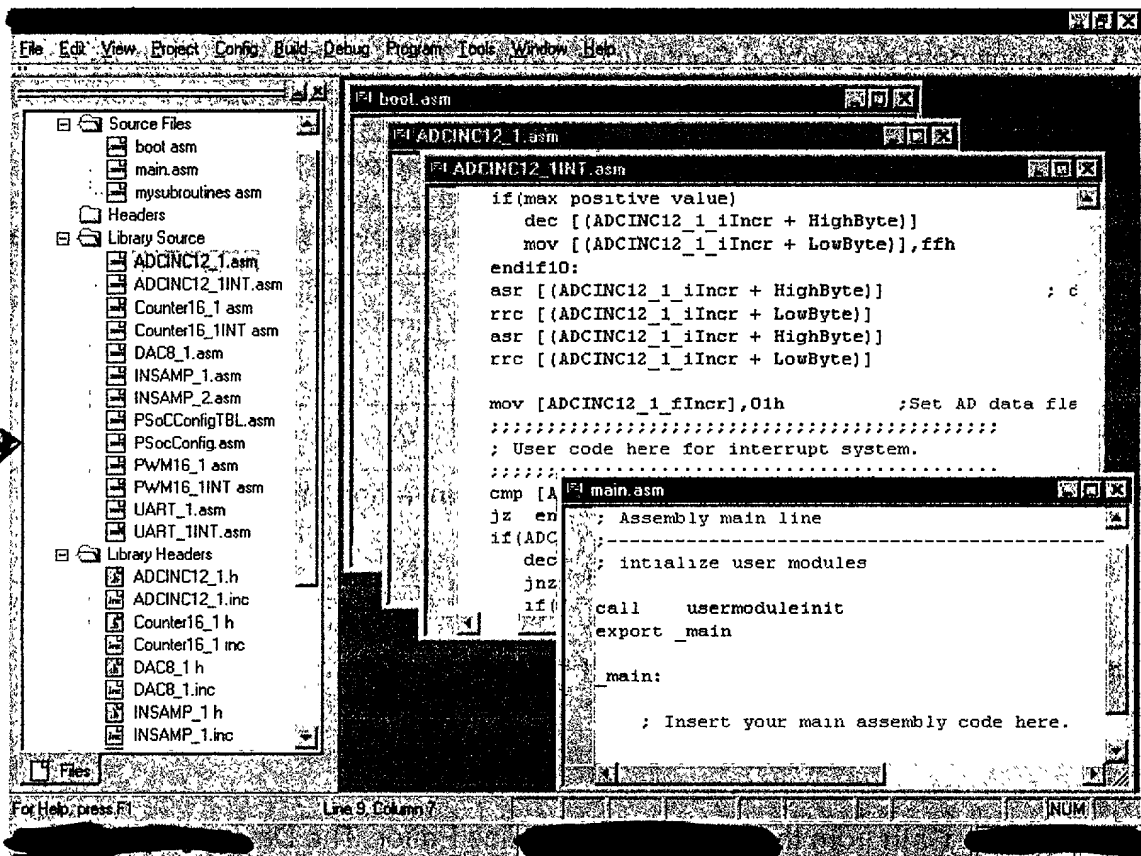


Fig. 1D

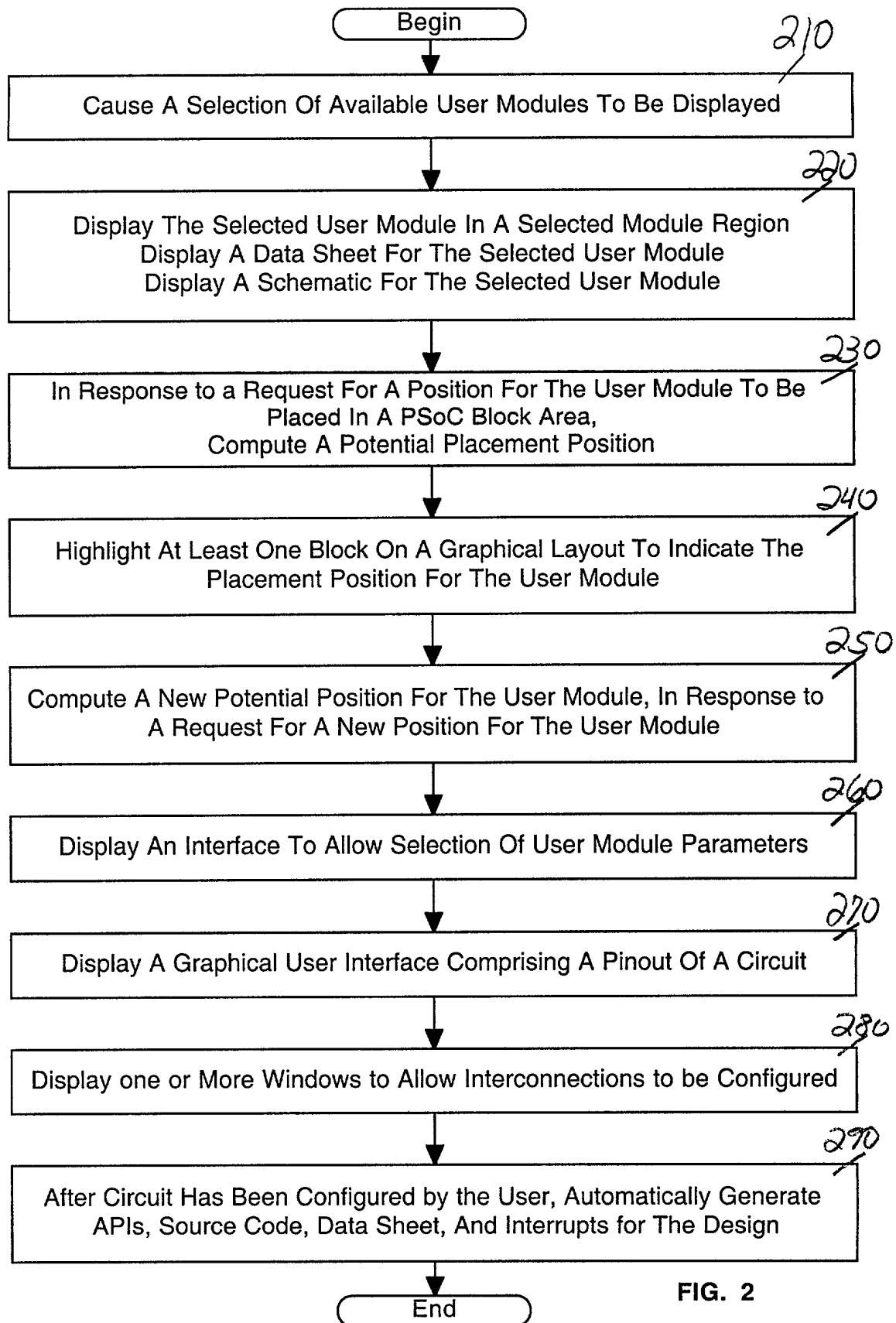
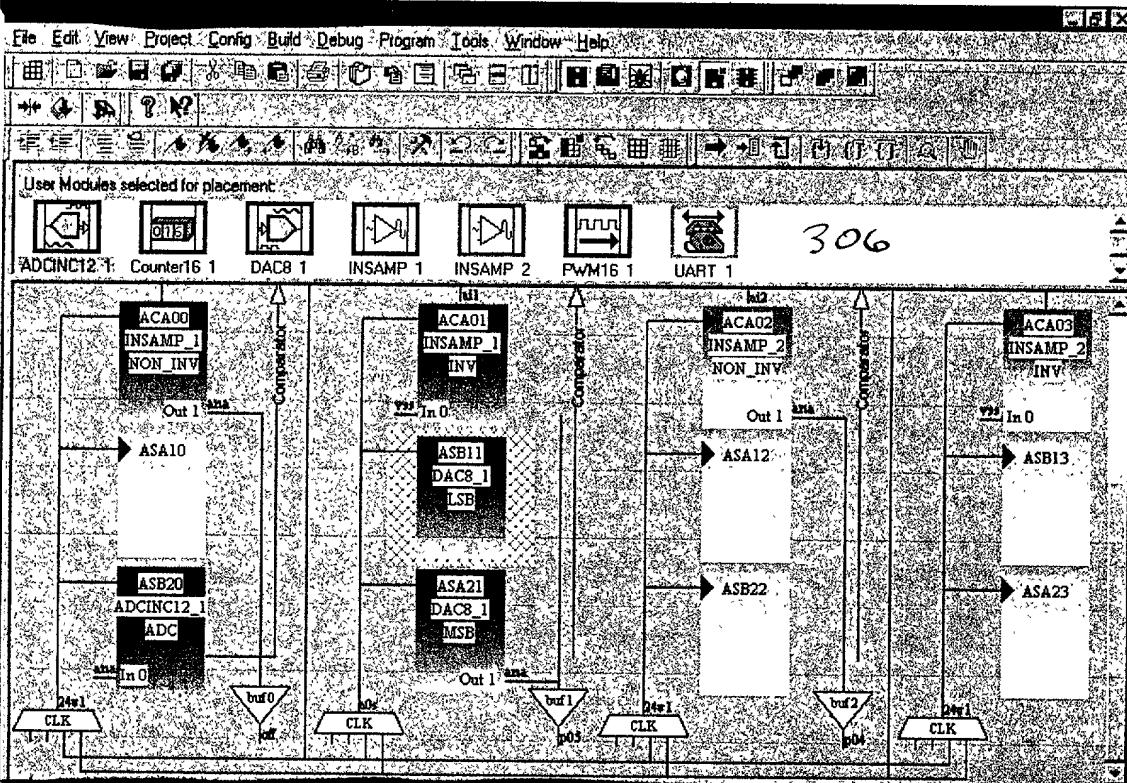


FIG. 2

	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431	2432	2
--	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	---



20250924 14:55:50

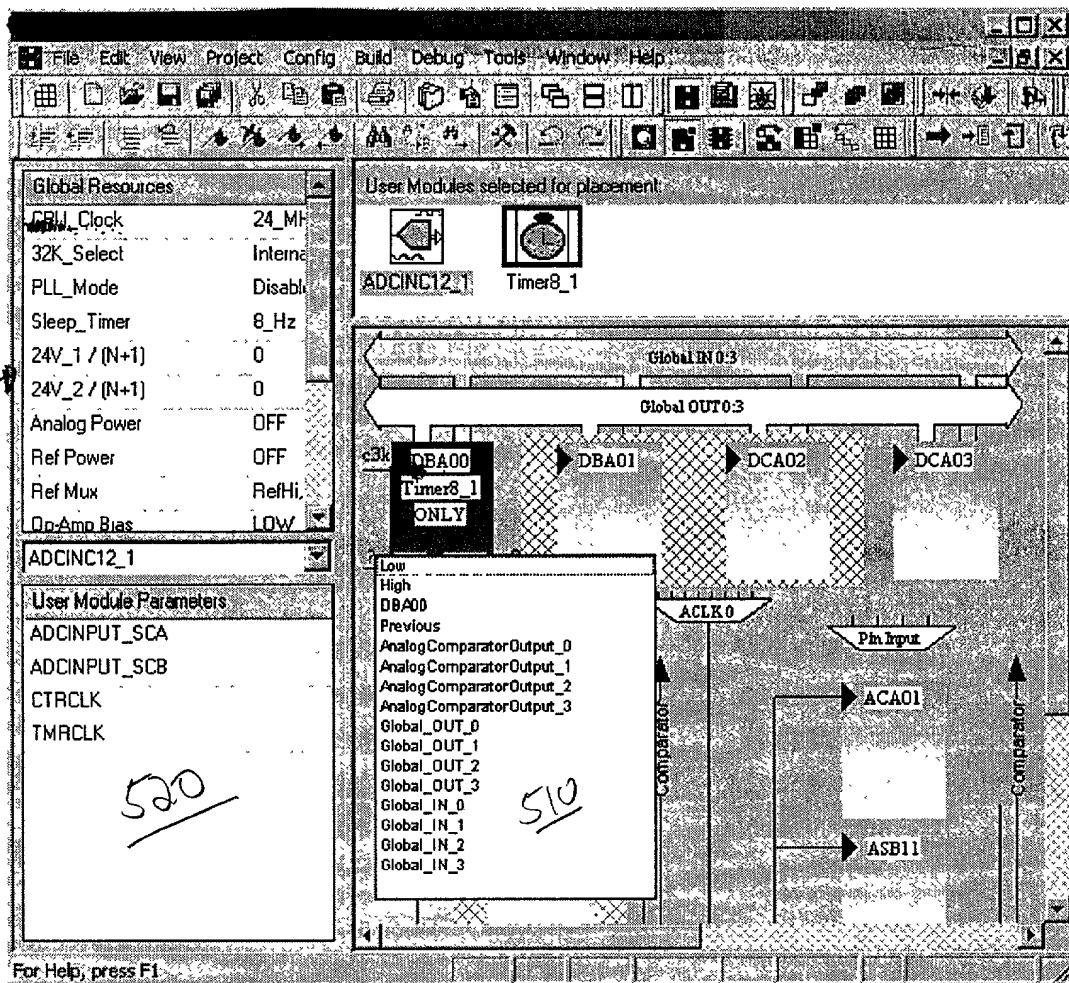


Fig. 4

09565570-44504

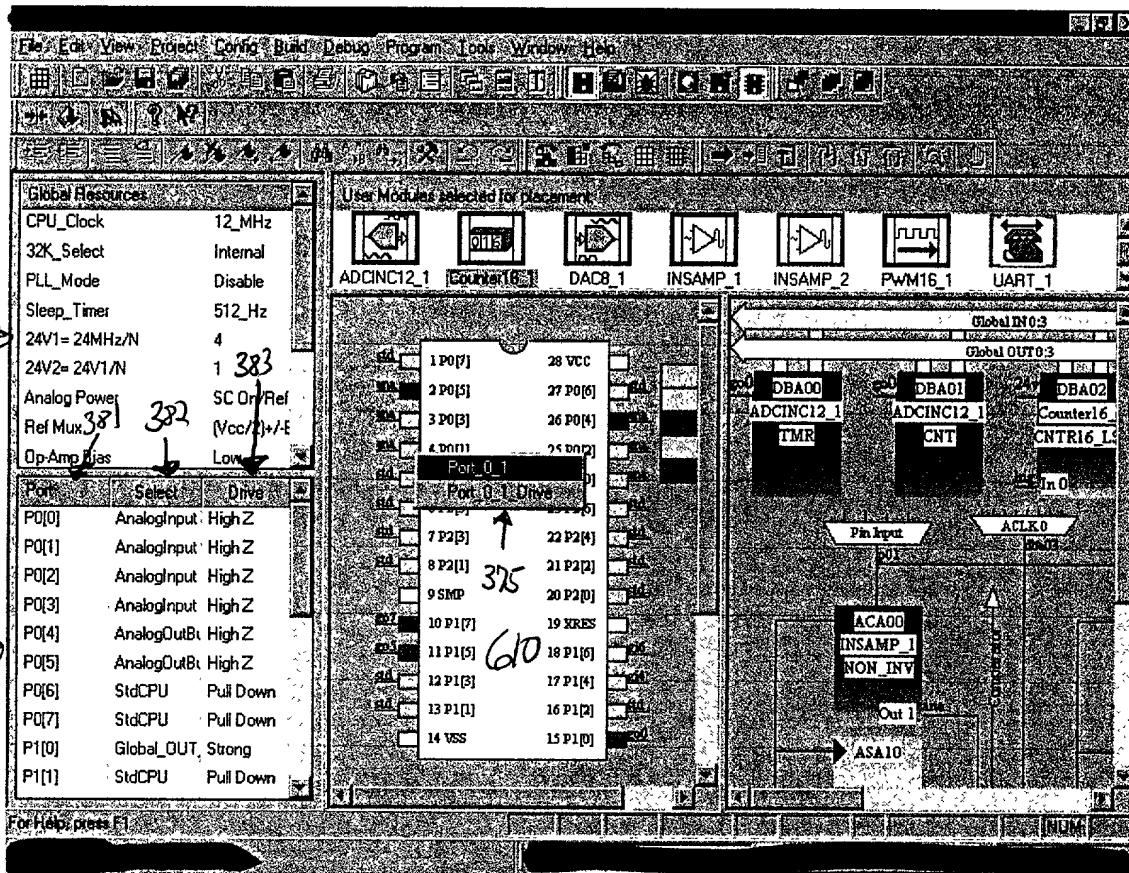


Fig. 54

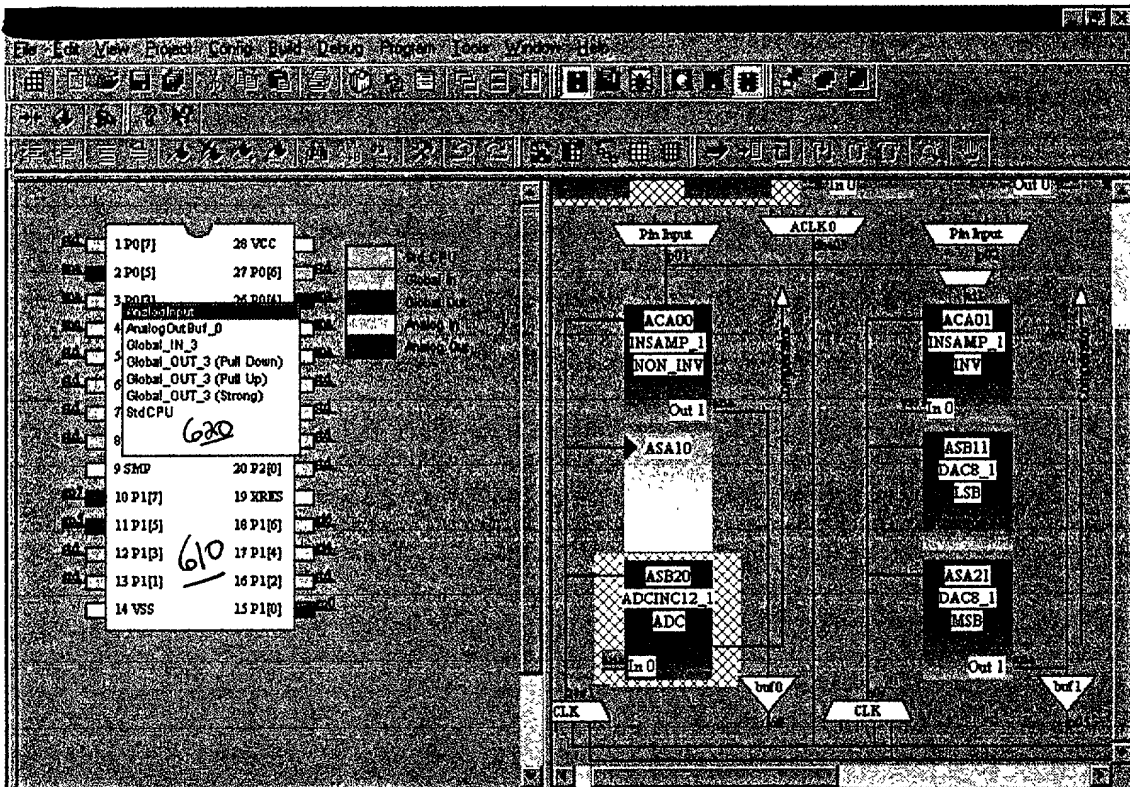


Fig. 5B

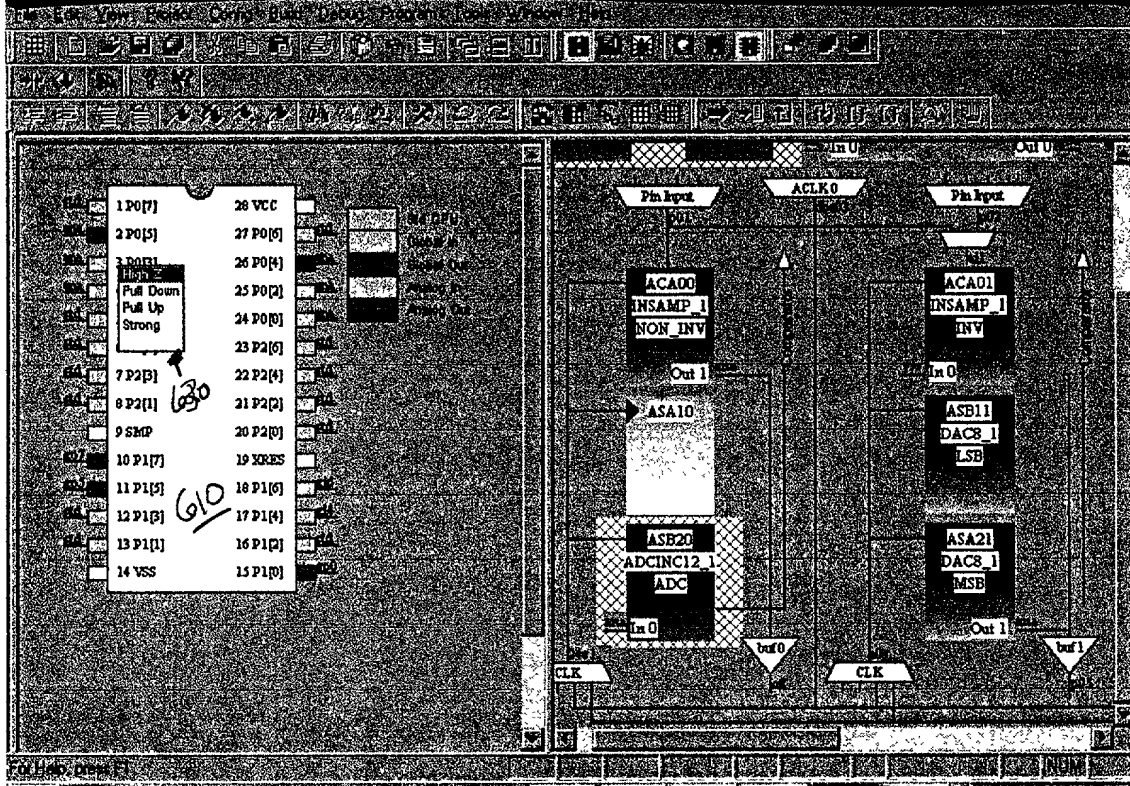


Fig. 5C

00555570 77504

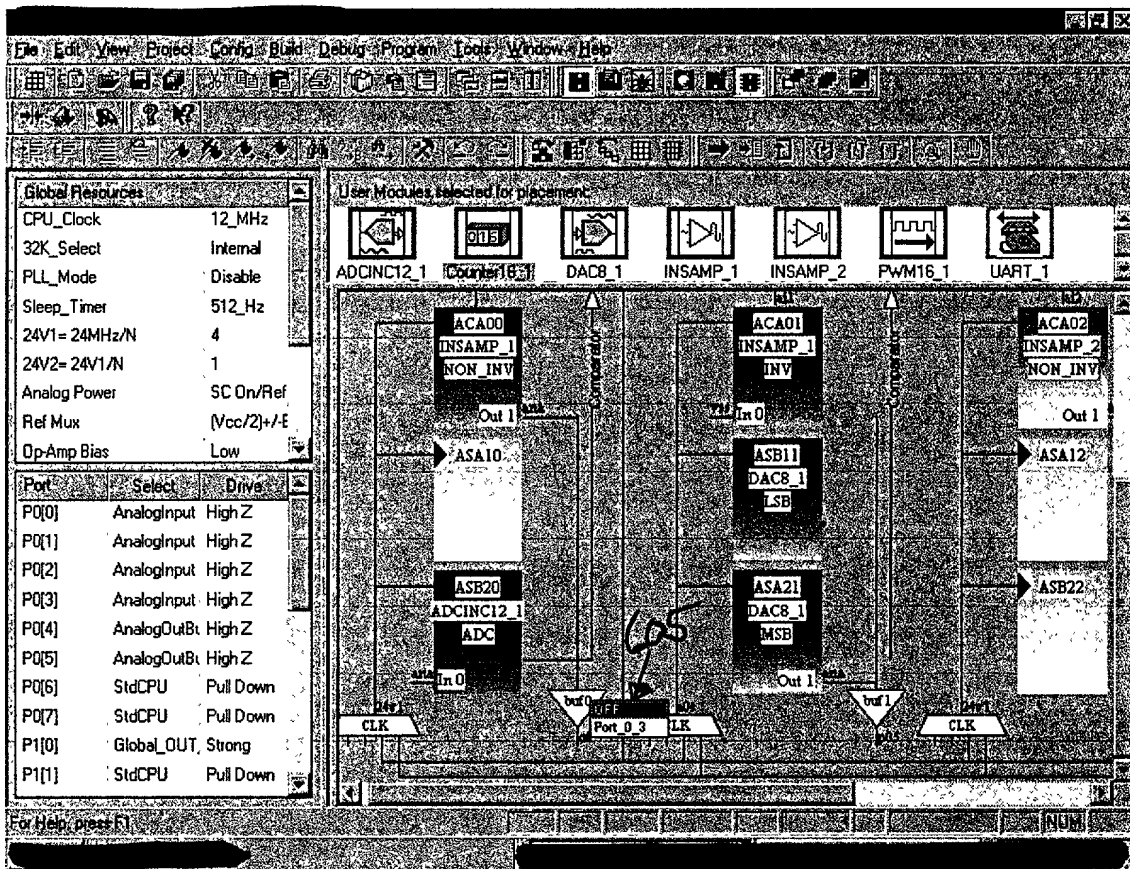


Fig. 6A

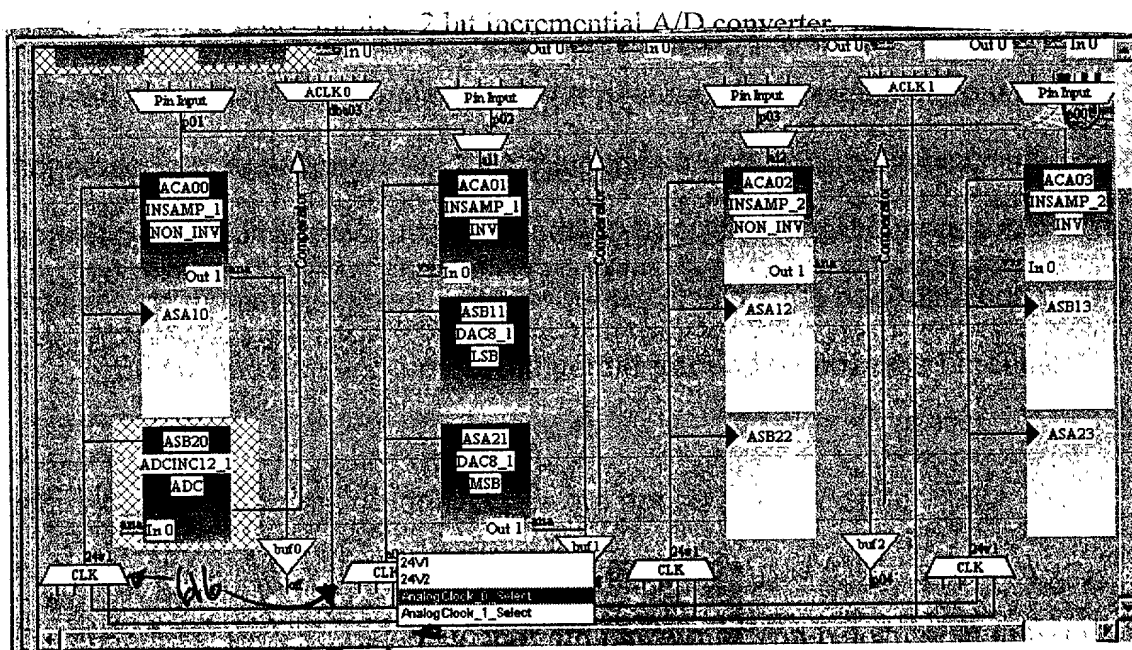


Fig. 6B


```

configtbl.asm
; Personalization tables
export LoadConfigTBL_project_Bank1
export LoadConfigTBL_project_Bank0
LoadConfigTBL_project_Bank1:
; Global Register values
    db            61h, 03h            ; AnalogClockSelect register
    db            60h, 08h            ; AnalogColumnClockSelect register
    db            62h, 30h            ; AnalogIOControl register
    db            63h, 00h            ; AnalogModulatorControl register
    db            e1h, 30h            ; OscillatorControl_1 register
    db            00h, 00h            ; Port_0_DriveMode_0 register
    db            01h, 3fh            ; Port_0_DriveMode_1 register
    db            04h, a1h            ; Port_1_DriveMode_0 register
    db            05h, 50h            ; Port_1_DriveMode_1 register
    db            08h, 00h            ; Port_2_DriveMode_0 register
    db            09h, 00h            ; Port_2_DriveMode_1 register
    db            0ch, 00h            ; Port_3_DriveMode_0 register
    db            0dh, 00h            ; Port_3_DriveMode_1 register
    db            10h, 00h            ; Port_4_DriveMode_0 register
    db            11h, 00h            ; Port_4_DriveMode_1 register
    db            14h, 00h            ; Port_5_DriveMode_0 register
    db            15h, 00h            ; Port_5_DriveMode_1 register
    db            e3h, 84h            ; VoltageMonitorControl register
; Instance name ADCINC12_1, User Module ADCINC12
;   Instance name ADCINC12_1, Block Name ADC(ASB20)
        db            90h, 90h            ;ADCINC12_1_AtoDcr0
        db            91h, 60h            ;ADCINC12_1_AtoDcr1
        db            92h, 60h            ;ADCINC12_1_AtoDcr2
        db            93h, f0h            ;ADCINC12_1_AtoDcr3
;   Instance name ADCINC12_1, Block Name CNT(DBA01)
        db            24h, 21h            ;ADCINC12_1_CounterFN
        db            25h, 48h            ;ADCINC12_1_CounterSL
        db            26h, 00h            ;ADCINC12_1_CounterOS
;   Instance name ADCINC12_1, Block Name TMR(DBA00)
        db            20h, 20h            ;ADCINC12_1_TimerFN
        db            21h, 18h            ;ADCINC12_1_TimerSL
        db            22h, 00h            ;ADCINC12_1_TimerOS
; Instance name Counter16_1, User Module Counter16
;   Instance name Counter16_1, Block Name CNTR16_LSB(DBA02)
        db            28h, 01h            ;Counter16_1_FUNC_LSB_REG
        db            29h, 16h            ;Counter16_1_INPUT_LSB_REG
        db            2ah, 00h            ;Counter16_1_OUTPUT_LSB_REG
;   Instance name Counter16_1, Block Name CNTR16_MSB(DBA03)
        db            2ch, 21h            ;Counter16_1_FUNC_MSB_REG
        db            2dh, 36h            ;Counter16_1_INPUT_MSB_REG
        db            2eh, 04h            ;Counter16_1_OUTPUT_MSB_REG
; Instance name DAC8_1, User Module DAC8
;   Instance name DAC8_1, Block Name LSB(ASB11)
        db            84h, 80h            ;DAC8_1_LSB_CR0
        db            85h, 80h            ;DAC8_1_LSB_CR1
        db            86h, 20h            ;DAC8_1_LSB_CR2
        db            87h, 30h            ;DAC8_1_LSB_CR3
;   Instance name DAC8_1, Block Name MSB(ASA21)
        db            94h, a0h            ;DAC8_1_MSB_CR0

```

Fig. 7A

(Continued)

```

db          95h, 41h      ;DAC8_1_MSB_CR1
db          96h, a0h      ;DAC8_1_MSB_CR2
db          97h, 30h      ;DAC8_1_MSB_CR3
; Instance name INSAMP_1, User Module INSAMP
;   Instance name INSAMP_1, Block Name INV(ACA01)
db          75h, beh      ;INSAMP_1_INV_CR0
db          76h, 21h      ;INSAMP_1_INV_CR1
db          77h, 20h      ;INSAMP_1_INV_CR2
;   Instance name INSAMP_1, Block Name NON_INV(ACA00)
db          71h, 3ch      ;INSAMP_1_NON_INV_CR0
db          72h, a1h      ;INSAMP_1_NON_INV_CR1
db          73h, 20h      ;INSAMP_1_NON_INV_CR2
; Instance name INSAMP_2, User Module INSAMP
;   Instance name INSAMP_2, Block Name INV(ACA03)
db          7dh, ceh      ;INSAMP_2_INV_CR0
db          7eh, 21h      ;INSAMP_2_INV_CR1
db          7fh, 20h      ;INSAMP_2_INV_CR2
;   Instance name INSAMP_2, Block Name NON_INV(ACA02)
db          79h, 2ch      ;INSAMP_2_NON_INV_CR0
db          7ah, a1h      ;INSAMP_2_NON_INV_CR1
db          7bh, 20h      ;INSAMP_2_NON_INV_CR2
; Instance name PWM16_1, User Module PWM16
;   Instance name PWM16_1, Block Name PWM16_LSB(DCA04)
db          30h, 01h      ;PWM16_1_FUNC_LSB_REG
db          31h, c4h      ;PWM16_1_INPUT_LSB_REG
db          32h, 00h      ;PWM16_1_OUTPUT_LSB_REG
;   Instance name PWM16_1, Block Name PWM16_MSB(DCA05)
db          34h, 21h      ;PWM16_1_FUNC_MSB_REG
db          35h, 34h      ;PWM16_1_INPUT_MSB_REG
db          36h, 05h      ;PWM16_1_OUTPUT_MSB_REG
; Instance name UART_1, User Module UART
;   Instance name UART_1, Block Name RX(DCA07)
db          3ch, 05h      ;UART_1_RX_FUNC_REG
db          3dh, e1h      ;UART_1_RX_INPUT_REG
db          3eh, 00h      ;UART_1_RX_OUTPUT_REG
;   Instance name UART_1, Block Name TX(DCA06)
db          38h, 0dh      ;UART_1_TX_FUNC_REG
db          39h, 01h      ;UART_1_TX_INPUT_REG
db          3ah, 07h      ;UART_1_TX_OUTPUT_REG
db          ffh
LoadConfigTBL_project_Bank0:
; Global Register values
db          60h, 14h      ; AnalogColumnInputSelect register
db          63h, 05h      ; AnalogReferenceControl register
db          65h, 00h      ; AnalogSyncControl register
db          e6h, 00h      ; DecimatorControl register
db          02h, 00h      ; Port_0_Bypass register
db          06h, f1h      ; Port_1_Bypass register
db          0ah, 00h      ; Port_2_Bypass register
db          0eh, 00h      ; Port_3_Bypass register
db          12h, 00h      ; Port_4_Bypass register
db          16h, 00h      ; Port_5_Bypass register
; Instance name ADCINC12_1, User Module ADCINC12
;   Instance name ADCINC12_1, Block Name ADC(ASB20)
;   Instance name ADCINC12_1, Block Name CNT(DBA01)

```

Fig. 7B

(Continued)

```
db          27h, 00h      ;ADCINC12_1_CounterCR0
db          25h, 00h      ;ADCINC12_1_CounterDR1
db          26h, 00h      ;ADCINC12_1_CounterDR2
; Instance name ADCINC12_1, Block Name TMR(DBA00)
db          23h, 00h      ;ADCINC12_1_TimerCR0
db          21h, 00h      ;ADCINC12_1_TimerDR1
db          22h, 00h      ;ADCINC12_1_TimerDR2
; Instance name Counter16_1, User Module Counter16
; Instance name Counter16_1, Block Name CNTR16_LSB(DBA02)
db          2bh, 00h      ;Counter16_1_CONTROL_LSB_REG
db          29h, 80h      ;Counter16_1_PERIOD_LSB_REG
db          2ah, 64h      ;Counter16_1_COMPARE_LSB_REG
; Instance name Counter16_1, Block Name CNTR16_MSB(DBA03)
db          2fh, 00h      ;Counter16_1_CONTROL_MSB_REG
db          2dh, 00h      ;Counter16_1_PERIOD_MSB_REG
db          2eh, 00h      ;Counter16_1_COMPARE_MSB_REG
; Instance name DAC8_1, User Module DAC8
; Instance name DAC8_1, Block Name LSB(ASB11)
; Instance name DAC8_1, Block Name MSB(ASA21)
; Instance name INSAMP_1, User Module INSAMP
; Instance name INSAMP_1, Block Name INV(ACA01)
; Instance name INSAMP_1, Block Name NON_INV(ACA00)
; Instance name INSAMP_2, User Module INSAMP
; Instance name INSAMP_2, Block Name INV(ACA03)
; Instance name INSAMP_2, Block Name NON_INV(ACA02)
; Instance name PWM16_1, User Module PWM16
; Instance name PWM16_1, Block Name PWM16_LSB(DCA04)
db          33h, 00h      ;PWM16_1_CONTROL_LSB_REG
db          31h, 37h      ;PWM16_1_PERIOD_LSB_REG
db          32h, 64h      ;PWM16_1_PWDITH_LSB_REG
; Instance name PWM16_1, Block Name PWM16_MSB(DCA05)
db          37h, 00h      ;PWM16_1_CONTROL_MSB_REG
db          35h, 00h      ;PWM16_1_PERIOD_MSB_REG
db          36h, 00h      ;PWM16_1_PWDITH_MSG_REG
; Instance name UART_1, User Module UART
; Instance name UART_1, Block Name RX(DCA07)
db          3fh, 00h      ;UART_1_RX_CONTROL_REG
db          3dh, 00h      ;UART_1_
db          3eh, 00h      ;UART_1_RX_BUFFER_REG
; Instance name UART_1, Block Name TX(DCA06)
db          3bh, 00h      ;UART_1_TX_CONTROL_REG
db          39h, 00h      ;UART_1_TX_BUFFER_REG
db          3ah, 00h      ;UART_1_
db          ffh
```

; PSoC Configuration file trailer PsocConfig.asm

Fig. 7C


```

; PSoCConfig.asm
;
; This file is generated by the Device Editor on Application Generation.
; It contains code which loads the configuration data table generated in
; the file PSoCConfigTBL.asm
;

export LoadConfigInit
export _LoadConfigInit
export LoadConfig_project
export _LoadConfig_project

FLAG_CFG_MASK:          equ    10h                ;M8C flag register REG address bit
mask
END_CONFIG_TABLE:      equ    ffh                ;end of config table indicator

_LoadConfigInit:
LoadConfigInit:
    lcall    LoadConfig_project

    ret

;
; Load Configuration project
;
_LoadConfig_project:
LoadConfig_project:
    or          F, FLAG_CFG_MASK                ;set for
bank 1
    mov         A, >LoadConfigTBL_project_Bank1 ;load bank 1 table
    mov         X, <LoadConfigTBL_project_Bank1
    call        LoadConfig                      ;load the
bank 1 values
    and         F, ~FLAG_CFG_MASK              ;switch
to bank 0
    mov         A, >LoadConfigTBL_project_Bank0 ;load bank 0 table
    mov         X, <LoadConfigTBL_project_Bank0
    call        LoadConfig                      ;load the
bank 0 values
    ret

;
; LoadConfig
;
; This function is not exported. It assumes that the address of the table
; to be loaded is contained in the X and A registers as if a romx instruction

```

Fig. 8A

(continued)

; is the next instruction to be executed, i.e. lower address in X and upper
; address in A. There is no return value.
;

LoadConfig:

LoadConfigLp:

```

    push    X                ;save config table address on stack
    push    A
    romx                    ;load config address
    cmp     A, END_CONFIG_TABLE ;check for end of table
    jz      EndLoadConfig    ;if so, end of load
    mov     X, SP            ;save the address away
    mov     [X], A
    pop     A                ;retrieve the table address
    pop     X
    inc     X                ;advance to the data byte
    jnc     NoOverflow1      ;check for overflow
    inc     A                ;if so, increment MSB

```

NoOverflow1:

```

    push    X                ;save the config table address
again
    push    A
    romx                    ;load the config data
    mov     X, SP            ;retrieve the config address
    mov     X, [X]
    mov     reg[X], A        ;write the config data
    pop     A                ;retrieve the table address
    pop     X
    inc     X                ;advance to the next

```

address

```

    jnc     NoOverflow2      ;check for overflow
    inc     A                ;if so, increment MSB

```

NoOverflow2:

```

    jmp     LoadConfigLp    ;loop back

```

EndLoadConfig:

```

    pop     A                ;clean up the stack
    pop     A
    ret

```

Fig. 8B

```

..*****
;;
..*****
;;
;; ADCINC12.asm
;;
;;
;; Assembler source for the 12 bit Incremental
;A/D converter.
;;
..*****
;;
..*****
;;

export ADCINC12_1_Start
export _ADCINC12_1_Start
export ADCINC12_1_SetPower
export _ADCINC12_1_SetPower
export ADCINC12_1_Stop
export _ADCINC12_1_Stop
export ADCINC12_1_GetSamples
export _ADCINC12_1_GetSamples
export ADCINC12_1_StopAD
export _ADCINC12_1_StopAD
export ADCINC12_1_flsData
export _ADCINC12_1_flsData
export ADCINC12_1_iGetData
export _ADCINC12_1_iGetData
export ADCINC12_1_ClearFlag
export _ADCINC12_1_ClearFlag

include "ADCINC12_1.inc"
include "m8c.inc"

LowByte: equ 1
HighByte: equ 0

;;-----
;; Start:
;; SetPower:
;; Applies power setting to the module's analog
;;PSoc block.
;; INPUTS: A contains the power setting
;; OUTPUTS: None.
;;-----
ADCINC12_1_Start:
_ADCINC12_1_Start:
ADCINC12_1_SetPower:
_ADCINC12_1_SetPower:
    and A,03h
    or A,f0h

```

```

mov reg[ADCINC12_1_AtoDcr3],A
ret

```

```

;;-----
;; Stop:
;; SetPower:
;; Removes power from the module's analog
;;PSoc block.
;; INPUTS: None.
;; OUTPUTS: None.
;;-----
ADCINC12_1_Stop:
_ADCINC12_1_Stop:
    and reg[ADCINC12_1_AtoDcr3], ~03h
    ret

;;-----
;; Get_Samples:
;; SetPower:
;; Starts the A/D convertor and will place data in
;;memory. A flag
;; is set whenever a new data value is available.
;; INPUTS: A passes the number of samples (0
;;is continuous).
;; OUTPUTS: None.
;;-----
ADCINC12_1_GetSamples:
_ADCINC12_1_GetSamples:
    mov [ADCINC12_1_bIncrC],A        ;number
;of samples
    or reg[INT_MSK1],(ADCINC12_1_TimerMask |
ADCINC12_1_CounterMask )
                                ;Enable both interrupts
    mov [ADCINC12_1_cTimerU],0      ;Force the
;Timer to do one cycle of rest
    or reg[ADCINC12_1_AtoDcr3],10h ;force the
;Integrator into reset
    mov [ADCINC12_1_cCounterU],ffh ;Initialize
;Counter

    mov reg[ADCINC12_1_TimerDR1],ffh
    mov reg[ADCINC12_1_CounterDR1],ffh
    mov reg[ADCINC12_1_TimerCR0],01h ;enable
;the Timer
    mov [ADCINC12_1_flgcr],00h      ;A/D Data
;Ready Flag is reset
    ret

```

(Continued)

```

;;-----
;; StopAD:
;; Completely shuts down the A/D in an orderly
;;manner. Both the
;; Timer and COUNTER interrupts are disabled.
;; INPUTS: None.
;; OUTPUTS: None.
;;-----
ADCINC12_1_StopAD:
_ADCINC12_1_StopAD:
    mov reg[ADCINC12_1_TimerCR0],00h
;disable the Timer
    mov reg[ADCINC12_1_CounterCR0],00h
;disable the Counter
    nop
    nop
    and
reg[INT_MSK1],~(ADCINC12_1_TimerMask |
ADCINC12_1_CounterMask)
                ;Disable both

;;interrupts
    or reg[ADCINC12_1_AtoDcr3],10h    ;reset
;;Integrator
    ret
;;-----
;; flsData:
;; Returns the status of the A/D Data
;; is set whenever a new data value is available.
;; INPUTS: None.
;; OUTPUTS: A returned data status A =: 0 no
;;data available
;;
                !=: 0 data available.
;;-----
ADCINC12_1_flsData:
_ADCINC12_1_flsData:
    mov A,[ADCINC12_1_flgcr]
    ret
;;-----
;; iGetData:
;; Returns the data from the A/D. Does not
;;check if data is
;; available.
;; is set whenever a new data value is available.
;; INPUTS: None.
;; OUTPUTS: X:A returns the A/D data value.
;;-----
ADCINC12_1_iGetData:

```

```

_ADCINC12_1_iGetData:
    mov X,[(ADCINC12_1_flgcr + HighByte)]
    mov A,[(ADCINC12_1_flgcr + LowByte)]
    ret

```

```

;;-----
;; ClearFlag:
;; clears the data ready flag.
;; INPUTS: None.
;; OUTPUTS: None.
;;-----
ADCINC12_1_ClearFlag:
_ADCINC12_1_ClearFlag:
    mov [ADCINC12_1_flgcr],00h
    ret

```

```

ADCINC12_1_API_End:

```

Fig. 9B

HEADER FILES //*****

//*****

//

// ADCINC12_1.h for the 12 bit incremental A/D converter

//

// C declarations for the ACDINC12 User Module.

//

//

//*****

//*****

#define ADCINC12_1_OFF 0

#define ADCINC12_1_LOWPOWER 1

#define ADCINC12_1_MEDPOWER 2

#define ADCINC12_1_HIGHPOWER 3

#pragma fastcall ADCINC12_1_Start

#pragma fastcall ADCINC12_1_SetPower

#pragma fastcall ADCINC12_1_GetSamples

#pragma fastcall ADCINC12_1_StopAD

#pragma fastcall ADCINC12_1_Stop

#pragma fastcall ADCINC12_1_flsData

#pragma fastcall ADCINC12_1_iGetData

#pragma fastcall ADCINC12_1_ClearFlag

extern void ADCINC12_1_Start(char power);

extern void ADCINC12_1_SetPower(char power);

extern void ADCINC12_1_GetSamples(char chout);

extern void ADCINC12_1_StopAD(void);

extern void ADCINC12_1_Stop(void);

extern char ADCINC12_1_flsData(void);

extern int ADCINC12_1_iGetData(void);

extern void ADCINC12_1_ClearFlag(void);

Fig. 10

```

..*****
;;
;;
;; ADCINC12_1.inc for the 12 bit incremental A/D converter
;;
;; Assembler declarations for the ACDINC12 User Module.
....
;;;
..*****
;;
..*****
;;

ADCINC12_1_AtoDcr0:   equ   90h
ADCINC12_1_AtoDcr1:   equ   91h
ADCINC12_1_AtoDcr2:   equ   92h
ADCINC12_1_AtoDcr3:   equ   93h
ADCINC12_1_CounterFN: equ   24h
ADCINC12_1_CounterSL: equ   25h
ADCINC12_1_CounterOS: equ   26h
ADCINC12_1_CounterDR0: equ   24h
ADCINC12_1_CounterDR1: equ   25h
ADCINC12_1_CounterDR2: equ   26h
ADCINC12_1_CounterCR0: equ   27h
ADCINC12_1_TimerFN:   equ   20h
ADCINC12_1_TimerSL:   equ   21h
ADCINC12_1_TimerOS:   equ   22h
ADCINC12_1_TimerDR0:  equ   20h
ADCINC12_1_TimerDR1:  equ   21h
ADCINC12_1_TimerDR2:  equ   22h
ADCINC12_1_TimerCR0:  equ   23h
ADCINC12_1_TimerMask: equ   01h
ADCINC12_1_CounterMask: equ 02h
ADCINC12_1_OFF:       equ   0
ADCINC12_1_LOWPOWER:  equ   1
ADCINC12_1_MEDPOWER:  equ   2
ADCINC12_1_HIGHPOWER: equ   3
ADCINC12_1_NUMBITS:   equ  12

```

Fig. 11

```

;*****
;;
;; ADCINC12int.asm
;;
;; Assembler source for interrupt routines the 12 bit Incremental
;; A/D Converter
;*****

export ADCINC12_1_CNT_INT
export ADCINC12_1_TMR_INT
include "ADCINC12_1.inc"
include "m8c.inc"

area bss(RAM)
    ADCINC12_1_cTimerU: BLK 1 ;The Upper byte of the Timer
    ADCINC12_1_cCounterU: BLK 1 ;The Upper byte of the Counter
    _ADCINC12_1_ilncr:
    ADCINC12_1_ilncr: BLK 2 ;A/D value
    _ADCINC12_1_flncr:
    ADCINC12_1_flncr: BLK 1 ;Data Valid Flag
    ADCINC12_1_blncrC: BLK 1 ;# of times to run A/D

area text(ROM,REL)

export ADCINC12_1_cTimerU
export ADCINC12_1_cCounterU
export _ADCINC12_1_ilncr
export ADCINC12_1_ilncr
export _ADCINC12_1_flncr
export ADCINC12_1_flncr
export ADCINC12_1_blncrC

LowByte: equ 1
HighByte: equ 0

;*****
;; CNT_INT:
;; Decrement the upper (software) half on the counter whenever the
;; lower (hardware) half of the counter underflows.
;; INPUTS: None.
;; OUTPUTS: None.
;*****
ADCINC12_1_CNT_INT:
    dec [ADCINC12_1_cCounterU]
    reti

;*****
;; TMR_INT:
;; This routine allows the counter to collect data for 64 timer cycles
;; This routine then holds the integrater in reset for one cycle while
;; the A/D value is calculated.
;; INPUTS: None.
;; OUTPUTS: None.
;*****
ADCINC12_1_TMR_INT:
    dec [ADCINC12_1_cTimerU]
; if(upper count >=0 )
    jc else1
    reti
else1:;(upper count decremented pass 0)
    tst reg[ADCINC12_1_AtoDcr3],10h ;to change when ice is fixed dbz
    jz else2

```

Fig 12A

(continued)

```

; if(A/D has been in reset mode)
    mov reg[ADCINC12_1_CounterCR0],01h ; Enable Counter
    and reg[ADCINC12_1_AtoDcr3],~10h ; Enable Analog Integrator
    mov [ADCINC12_1_cTimerU],((1<<(ADCINC12_1_NUMBITS - 6))-1)
                                ; This will be the real counter value

    reti

else2::(A/D has been in integrate mode)
    mov reg[ADCINC12_1_CounterCR0],00h ;disable counter
    or F,01h ;Enable the interrupts
    ; Good place to add code to switch inputs for multiplexed input to ADC
    ; Reset Integrator
    or reg[ADCINC12_1_AtoDcr3],10h
    mov [(ADCINC12_1_ilncr + LowByte)],ffh
    mov [(ADCINC12_1_ilncr + HighByte)],(ffh - (1<<(ADCINC12_1_NUMBITS - 7)))

;
    push A
    mov A, reg[ADCINC12_1_CounterDR0] ;read Counter
    mov A, reg[ADCINC12_1_CounterDR2] ;now you really read the data
    sub [(ADCINC12_1_ilncr + LowByte)],A
    mov A,[ADCINC12_1_cCounterU]
    sbb [(ADCINC12_1_ilncr + HighByte)],A
    pop A
    cmp [(ADCINC12_1_ilncr + HighByte)],(1<<(ADCINC12_1_NUMBITS - 7))
    jnz endif10
; if(max positive value)
    dec [(ADCINC12_1_ilncr + HighByte)]
    mov [(ADCINC12_1_ilncr + LowByte)],ffh
endif10:
    asr [(ADCINC12_1_ilncr + HighByte)] ; divide by 4
    rrc [(ADCINC12_1_ilncr + LowByte)]
    asr [(ADCINC12_1_ilncr + HighByte)]
    rrc [(ADCINC12_1_ilncr + LowByte)]

    mov [ADCINC12_1_flgcr],01h ;Set AD data flag
    ; User code here for interrupt system.
    cmp [ADCINC12_1_blnrc],00h
    jz endif3
; if(ADCINC12_1_blnrc is not zero)
    dec [ADCINC12_1_blnrc]
    jnz endif4
; if(ADCINC12_1_blnrc has decremented down to zero to 0)
    mov reg[ADCINC12_1_TimerCR0],00h ;disable the Timer
    mov reg[ADCINC12_1_CounterCR0],00h ;disable the Counter
    nop
    nop
    and reg[INT_MSK1],~(ADCINC12_1_TimerMask | ADCINC12_1_CounterMask)
                                ;Disable both interrupts
    or reg[ADCINC12_1_AtoDcr3],10h ;Reset Integrator
    reti
endif4::
endif3::
endif2::
    mov [ADCINC12_1_cTimerU],00h ;Set Timer for one cycle of reset
    mov [ADCINC12_1_cCounterU],ffh ;Set Counter hardware for easy enable
    mov reg[ADCINC12_1_CounterDR1],ffh
    reti
endif1::

```

ADCINC12_1_APIINT_END: A/D converter.

Fig. 12B

1300

```

;-----
; Interrupt Vector Table
;-----
;
; Interrupt vector table entries are 4 bytes long
; and contain the code
; that services the interrupt (or causes it to be
; serviced).
;
;-----

```

AREA TOP(ROM, ABS)

```

org 0          ; Reset Interrupt Vector
jmp __start    ; First instruction
;executed following a Reset

```

```

org 04h        ; Supply Monitor Interrupt
;Vector
// call void_handler
reti

```

```

org 08h        ; PSoC Block DBA00
;Interrupt Vector
1305 jmp ADCINC12_1_TMR_INT
reti

```

```

org 0Ch        ; PSoC Block DBA01
;Interrupt Vector
1305 jmp ADCINC12_1_CNT_INT
reti

```

```

org 10h        ; PSoC Block DBA02
;Interrupt Vector
// call void_handler
reti

```

```

org 14h        ; PSoC Block DBA03
;Interrupt Vector
jmp Counter16_1INT
reti

```

```

org 18h        ; PSoC Block DCA04
;Interrupt Vector
// call void_handler
reti

```

```

org 1Ch        ; PSoC Block DCA0
;Interrupt Vector
ljmp PWM16_1INT
reti

```

```

org 20h        ; PSoC Block DCA06
;Interrupt Vector
ljmp UART_1TX_INT
reti

```

```

org 24h        ; PSoC Block DCA07
;Interrupt Vector
1305 jmp UART_1RX_INT
reti

```

```

org 28h        ; Analog Column 0
;Interrupt Vector
// call void_handler
reti

```

```

org 2Ch        ; Analog Column 1
;Interrupt Vector
// call void_handler
reti

```

```

org 30h        ; Analog Column 2
;Interrupt Vector
// call void_handler
reti

```

```

org 34h        ; Analog Column 3
;Interrupt Vector
// call void_handler
reti

```

```

org 38h        ; GPIO Interrupt Vector
// call void_handler
reti

```

```

org 3Ch        ; Sleep Timer Interrupt
;Vector
jmp SleepTimerISR
reti

```

Fig. 13A

1351

<i>boof.asm</i> Interrupt Name	Data Sheet Interrupt Name	Type
_start	Reset	Fixed
Interrupt1	Supply Monitor	Fixed
Interrupt2	DBA00	PSoC Block
Interrupt3	DBA01	PSoC Block
Interrupt4	DBA02	PSoC Block
Interrupt5	DBA03	PSoC Block
Interrupt6	DCA04	PSoC Block
Interrupt7	DCA05	PSoC Block
Interrupt8	DCA06	PSoC Block
Interrupt9	DCA07	PSoC Block
Interrupt10	Analog Column 0	PSoC Block
Interrupt11	Analog Column 1	PSoC Block
Interrupt12	Analog Column 2	PSoC Block
Interrupt13	Analog Column 3	PSoC Block
Interrupt14	GPIO	Fixed
Interrupt15	Sleep Timer	Fixed

Fig. 13B

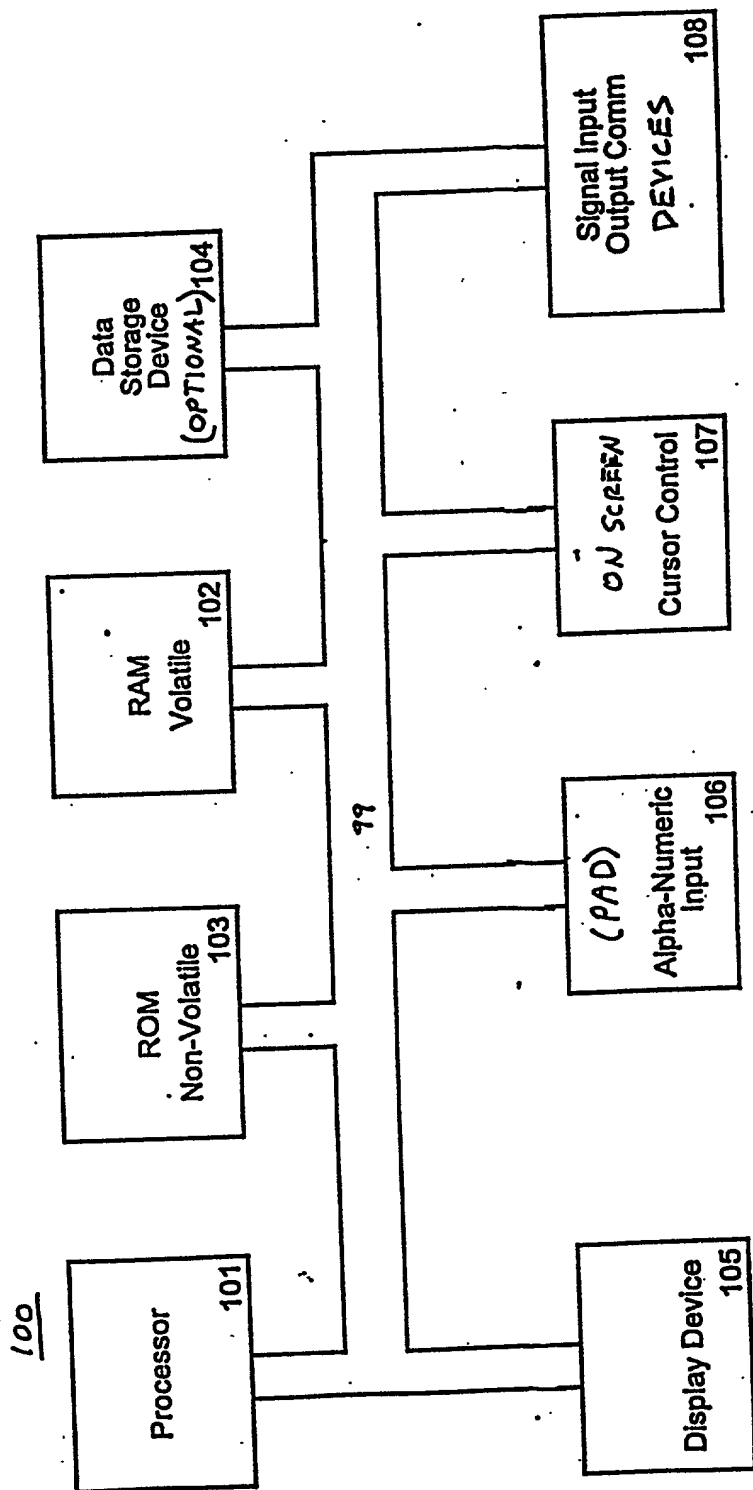


FIG. 14